# IMPROVED DIAGNOSTICS FOR TEST PROGRAM SETS WITH LARGE POSSIBLE FAULT DETECTS

David Yip
Support Systems Associates, Inc. - Lakehurst , New Jersey
908-323-2413
E-Mail: YIPDP@LAKEHURST.NAVY.MIL

Richard Lysakowski
Naval Air Warfare Center, Aircraft Division - Lakehurst, New Jersey
908-323-5236
E-Mail: LYSAKORP@LAKEHURST.NAVY.MIL

Joseph Mikolaj
Naval Surface Warfare Center - Crane, Indiana
812-854-5983
E-Mail: JOE@VRGNIA.NWSCC.SEA06.NAVY.MIL

# IMPROVED DIAGNOSTICS FOR TEST PROGRAM SETS WITH LARGE POSSIBLE FAULT DETECTS

David Yip
Richard Lysakowski
Joseph Mikolaj

## ABSTRACT

In a test engineering environment involving fault detection and fault isolation of units under test (UUT), a software system for simulating the performances of digital circuits is predicated on the use of weighting factors of definite and possible fault detects from a fault dictionary (a database of unique response output to a set of faults) and the calculation of a mismatch value for each set of faults. Inclusion of a high number of possible fault detects, used initially in reducing fault ambiguity size of faultsets, to calculate the mismatch value proved to be detrimental in isolating faults. This paper presents a new methodology in calculating the mismatch value for circuit designs with a high number of possible fault detects to improve diagnostic fault isolation accuracy.

## INTRODUCTION

This paper begins with the brief introduction to the processes of detecting and diagnosing faults on a UUT and then describes the inherent deficiency found in one of the tools used in isolating those faults. A definition of the essential elements of the process and the strategy for improvement is presented.

A fault dictionary is a diagnostic database generated by a Digital Automatic Test Program Generator (DATPG), a digital simulator, that aids in diagnosing faults during a functional test of a circuit board. A fault dictionary contains a list of possible faults for each scenario in a job. These faults are grouped into faultsets. A faultset is a set of faults, each of which will cause exactly the same output pins to fail at exactly the same patterns.

The digital simulator predicts faults by considering both definite detects and possible detects. A definite detect is defined as a fault that will produce a state difference at the primary output pins in response to Test Program Set (TPS) patterns. A possible detect is defined as a fault that produces an unknown state at the primary output pins when a fault-free UUT would produce a known state.

When running a functional test of an UUT on a automated tester, the run-time algorithm software compares the actual output pin failures to the fault dictionary. From the comparison, the simulation displays the most likely faultsets within the fault dictionary that best represents the failed devices that caused the output pin failure. A mismatch number is a number that reflects the likelihood that one of the faults in the fault set is the actual fault. The lower the mismatch number, the more likely it is that the faultset contains the actual fault. The lowest mismatch number, zero (0), is shown as an EXACT MATCH.

The mismatch value calculation is based on statistically defined weighting factors for penalizing the fault dictionary differences (that is, state differences found at the primary output pins in response to TPS patterns). A primary output difference predicted by the simulator that does not reveal itself on the automated tester counts as a nine point weighting factor. A primary output difference which manifest itself on the automated tester but was not predicted by the simulator also counts as a nine point weighting factor. A primary output difference possibly predicted by the simulator that does not reveal itself on the tester counts as a one point weighting factor.

## METHODOLOGY

The current calculation of the Fault Dictionary Mismatch Value embedded in the Teradyne VX2.02 software is based on summing the fault detects of definite and possible detects. Furthermore, definite and possible detects have different weighting factors and are predefined for calculating the mismatch value. This methodology may be inappropriate for circuits that have a high number of possible detects in the fault dictionary.

The problem arises when some UUTs are simulated with the inclusion of possible faults to aid in reducing the fault ambiguity size of faultsets. A high number of possible detects could cause the algorithm to overestimate the mismatch value for particular faultsets that may contain the actual fault. The algorithm would erroneously identify other faultsets that have lower mismatch values as having the actual fault.

For example, a UUT may have a faultset with a mismatch value of 21, (henceforth known as faultset A) and another faultset with a value of 23, (henceforth known as faultset B). Based on the mismatch value, the algorithm would select faultset A as having the actual fault since it has the lower mismatch value. However, the selection of this faultset may not be the best choice. If the number of fault detects for the faultset A was two definite detects (with a weighting factor of nine each, subtotaling to a value of 18), and three possible detects (with a weighting factor of one each, giving a grandtotal of 21) and faultset B has one definite detect and 14 possible detects, than the best choice should have been faultset B since the one definite detect is likely to be the actual fault.

## TEST STRATEGY

A study was conducted to develop methods to improve the accuracy of the fault dictionary. Several test strategies were suggested, which include: changing the weighting factors of definite and possible detects, calculating the mismatch value based on definite detects only, combining faultsets with equal mismatch values, and combining faultsets with equal mismatch values and sort by definite detects**.**

The study indicated that the best option was to sort the faultsets based on definite detects and in case of equal mismatch values to include the possible detects in the calculation. A prototype software was developed to implement this methodology in

calculating the mismatch value. Fault isolation accuracy showed significant improvement when using this new criteria.

This methodology was chosen to maintain the integrity of the weighting factor for the definite and possible detected faults since the weighting factors were statistically optimized in producing best scores for detected faults. Changing the weighting factors may produce erroneous scores that may not be the best representative selection of faultsets for fault isolation.

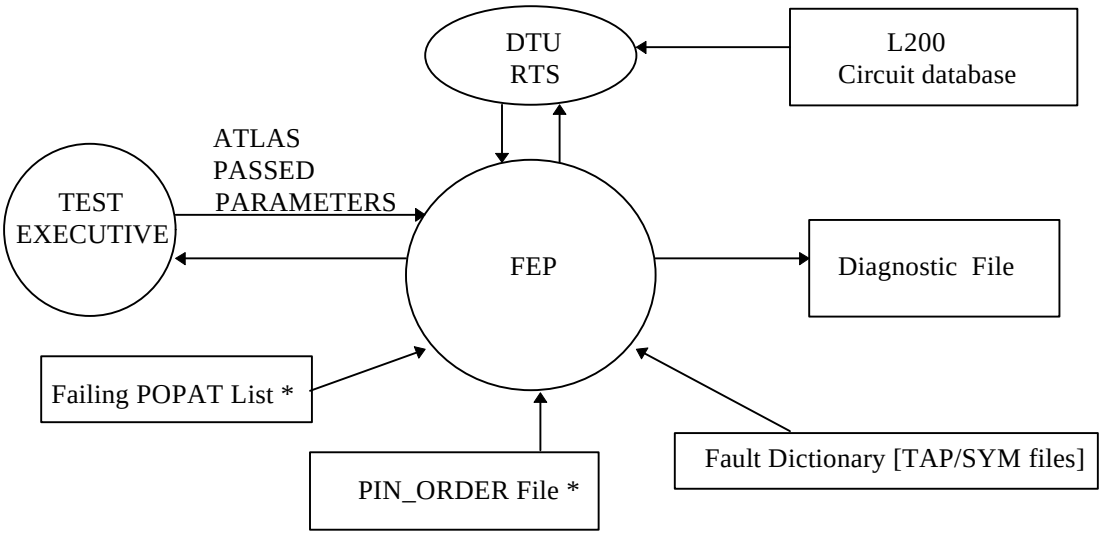
## IMPLEMENTATION

To ensure the integrity of the weighting factors for fault detects, the software was developed as a Functional Extension Program (FEP) to the VX2/DTU software. The Consolidated Automated Support System (CASS) station was used as the platform for executing the FEP. The FEP would calculate the mismatch value as opposed to generating the data via the VX2/DTU system software. The FEP would discard any possible detects as part of the calculation unless the resultant mismatch values for each of the faultsets were equal (in this case the software includes the possible detects to further isolate the fault).

The FEP is invoked by a CASS ATLAS (Abbreviated Test Language for All Systems) executive software. The FEP parses the Fault Dictionary file [.TAP or .SYM formatted file] for fault data and compares it to the failing POPAT (Primary Output Pattern) list. The routine then calculates the mismatch value based on the definite detects of the fault data. In addition, the circuit description file if available is parsed mainly to differentiate the drivers and receivers of failing pins. This is comparable to how the current Teradyne's VX2 software operates.

The parameters that are passed to the executive software are the Test-Failed boolean flag, the Mismatch Value score and the associated fault set data identified as the primary and alternate causes of failure. The input database files required are the aforementioned Failing POPAT list file, and the Fault Dictionary file. In addition, the FEP also requires the Pin Order data file which contains the ordered list of primary pins of the UUT. The database files that are generated for output are the Diagnostic file which contains the detailed description of the faults extracted from the faultsets with the lowest mismatch values and the Error file which contains any error messages.

The following diagram, figure 1, depicts the external data flow of the software that processes the passed parameters to and from other software units.




* Generated by the DTU RTS [Digital Test Unit / Run Time Software]

**Figure 1. External Data Flow Diagram**

The mismatch values are calculated for each faultset by extracting the data from the fault dictionary and tallying them based on definite detects only. In case of faultsets having equal mismatch values, possible detects are then used in the calculation to further isolate the faultset. In case of further equal mismatch values, the faultsets are combined into one faultset.

The selected faultset with the lowest mismatch value identifies the devices with the fault which will be determined as replaceable units. The determination of the replaceable units is based on the selected faultset and circuit database file.

## CONCLUSION

Unlike the current VX2/DTU software in calculating the mismatch value, the FEP derives its mismatch value by summing the results of definite detects only. As mentioned before, possible detects are only included in the calculation when equal mismatch values between faultsets are evident.

This paper has enumerated various test options, and proposed the aforementioned methodology for fault isolation. Conceptually, the methodology can be part of any test strategy in detecting and isolating faults. Full beta testing of the FEP successfully demonstrated the improvement in selecting correct faultsets.

## ACKNOWLEDGMENTS

The authors wish to thank the engineers at the Naval Surface Warfare Center (NSWC), Crane Indiana, for their assistance in performing beta testing. Their feedback and comments were extremely valuable in the software development process.

Special thanks to Joseph Mikolaj for conducting the initial study and developing the prototype software.